

Table of Contents

|  |  |  |
| --- | --- | --- |
| Sr no. | Content | Page Number |
| 1 | Introduction | 3 |
| 2 | Problem Definition | 4 |
| 3 | Module | 5 |
| 4 | Implementation (Complete Source Code) | 6 |
| 5 | Database | 11 |
| 6 | Explanation of the Code | 12 |
| 7 | Results | 13 |
| 8 | Scope of Improvements | 15 |
| 9 | Conclusion | 16 |
| 10 | References | 17 |

1] Introduction

1.1] OOPs

A paradigm of object-oriented programming gives a more object-related approach toward the design of software-thus, classes are conceptualized as instances and the application objects that encapsulate both data and behaviour. The fundamental principles of OOP are encapsulation, which binds together data and methods within an object with controlled access to some of its components; inheritance, which enables one class to inherit properties from another, thus enabling code reusability; polymorphism, which allows treating different objects as instances of a common superclass; and abstraction, where complex details of implementation are hidden and only essential features are exposed. All of these will improve reusability, scalability, and maintainability, thus forming the backbone of languages like Java.

1.2] Encryption

Encryption is a means of securing data. It transforms the data into an unreadable format, and access or deciphering is possible only for authorized parties. In other words, for messaging, encryption is important in ensuring that information communication involving sensitive matters is not accessed and thus to provide confidentiality and security. In this case, the messages are encrypted by transforming them into ciphertext. Only the ciphertext can be decrypted using a predefined key that has been given to the intended recipient. This would prevent interception, alteration, and data leaks especially with current digital platforms where most messages are transported over public networks. Encryption of messages ensures that the contents of the message are kept confidential, and thus authenticity and integrity are assured, and are therefore paramount for private as well as official communications.

2] Problem Definition

2.1] Motivation

Recent years, several high-profile breaches exposed millions of users' private messages on unsecured platforms, leading to financial loss, identity theft, and privacy invasion. This emphasizes the need for strong encryption to safeguard sensitive information. By developing a mini project on message encryption, we wanted to have basic idea of encryption, and aim to create secure communication tools, and contribute to a deeper understanding of protecting personal and professional conversations from such risks.

2.2] Problem Statement

The goal of this mini project is to design and implement a system that encrypts messages before they are transmitted, ensuring that only authorized recipients can decrypt and read the content. This project aims to develop a secure, user-friendly solution that provides confidentiality, data integrity, and authenticity for messaging systems, addressing the growing need for secure communication in both personal and professional contexts.

The project also covers how the sniffers will view the message if there is an encounter of breach in the transmission line.

3] Module

3.1] Encryption

Encrypts the data by comparing the character with the Database. It runs a query where the selected character is navigated in COL\_CH. It then finds the corresponding character from COL\_ENC and retrieves it.

3.2] Decryption

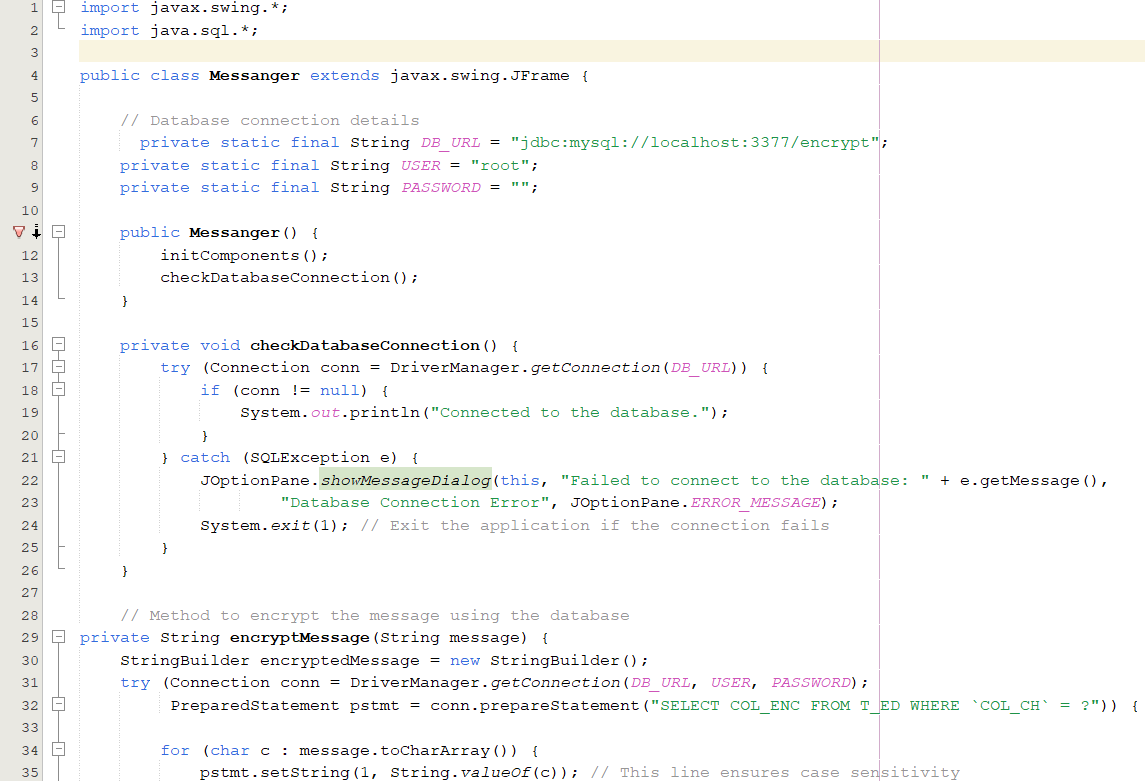
Decrypts the data by comparing the character with the Database. It runs a query where the selected character is navigated in COL\_ENC. It then finds the corresponding character from COL\_CH and retrieves it

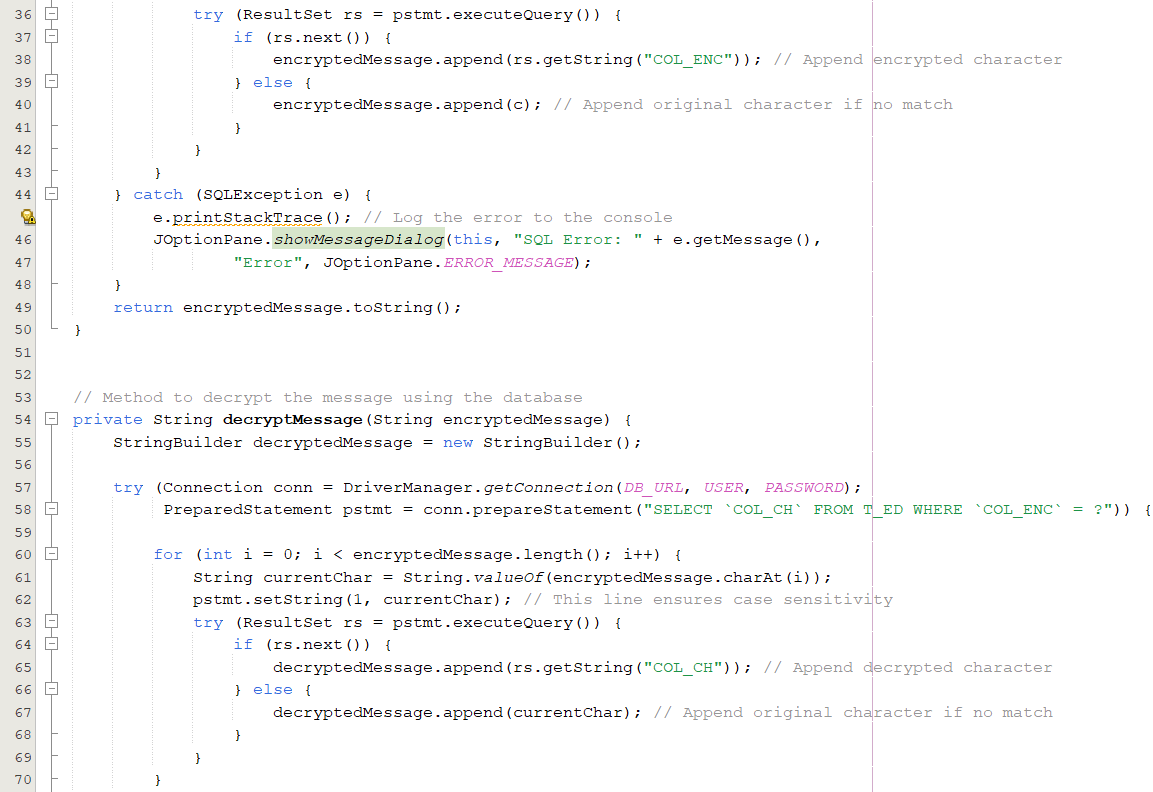
3.3] Send

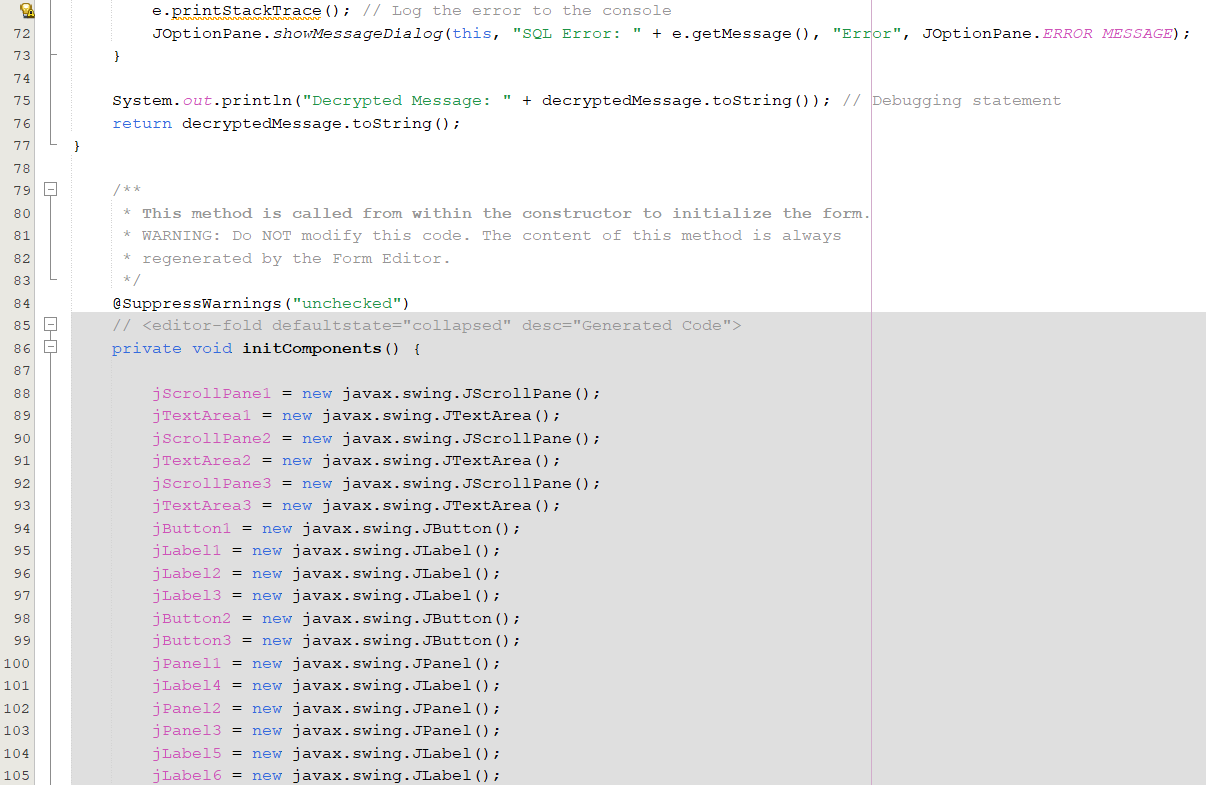
This performs both encryption and decryption of the data. This is the real-time simulation of how exactly the process runs

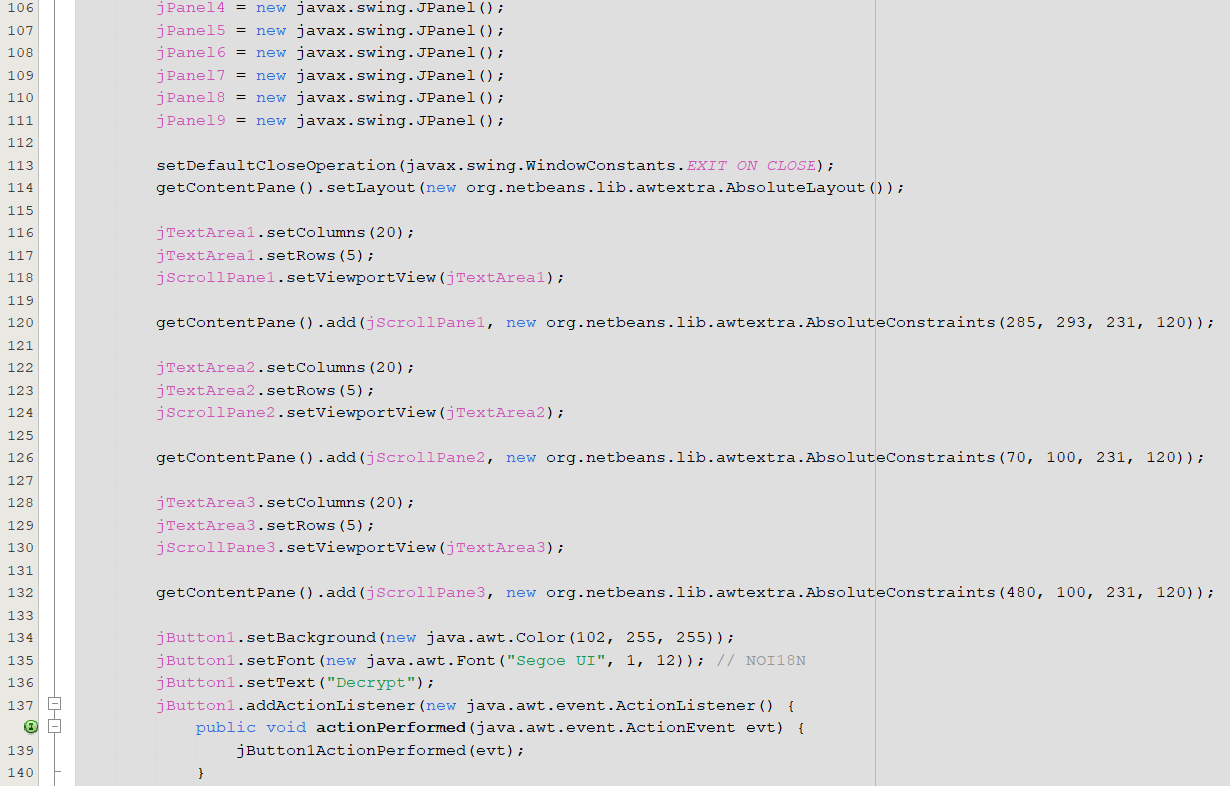
4] Implementation

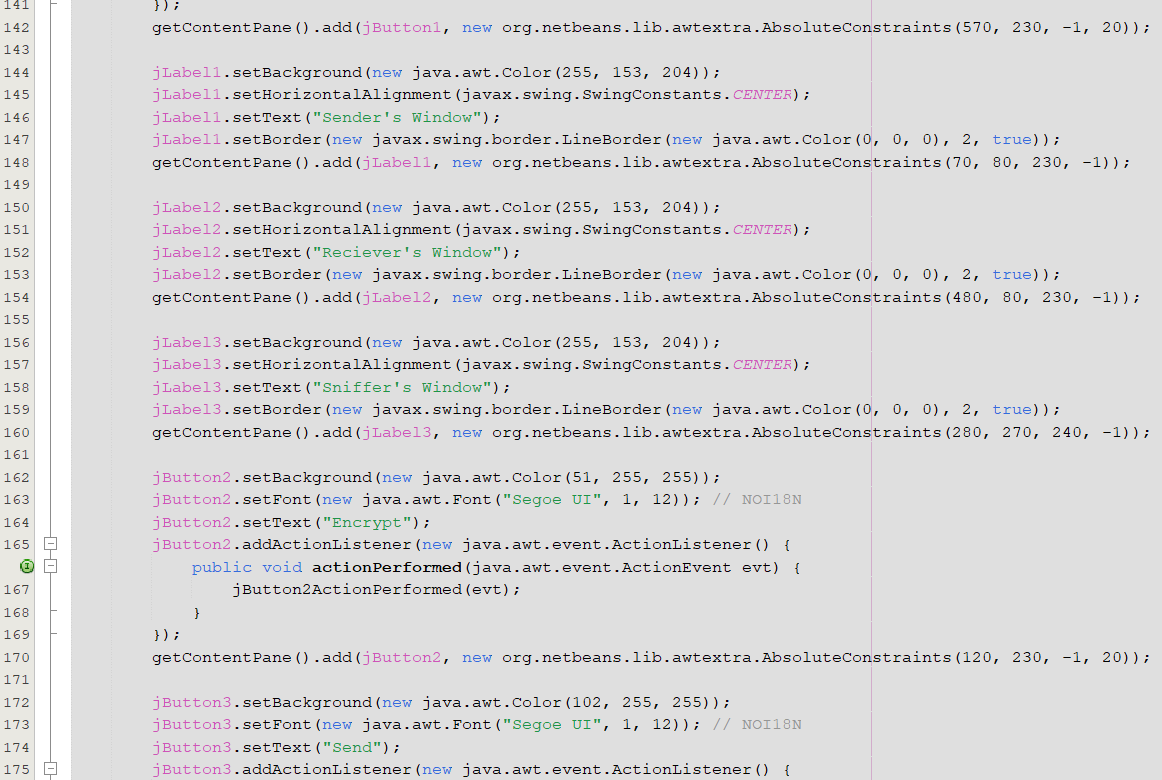
4.1] Source Code

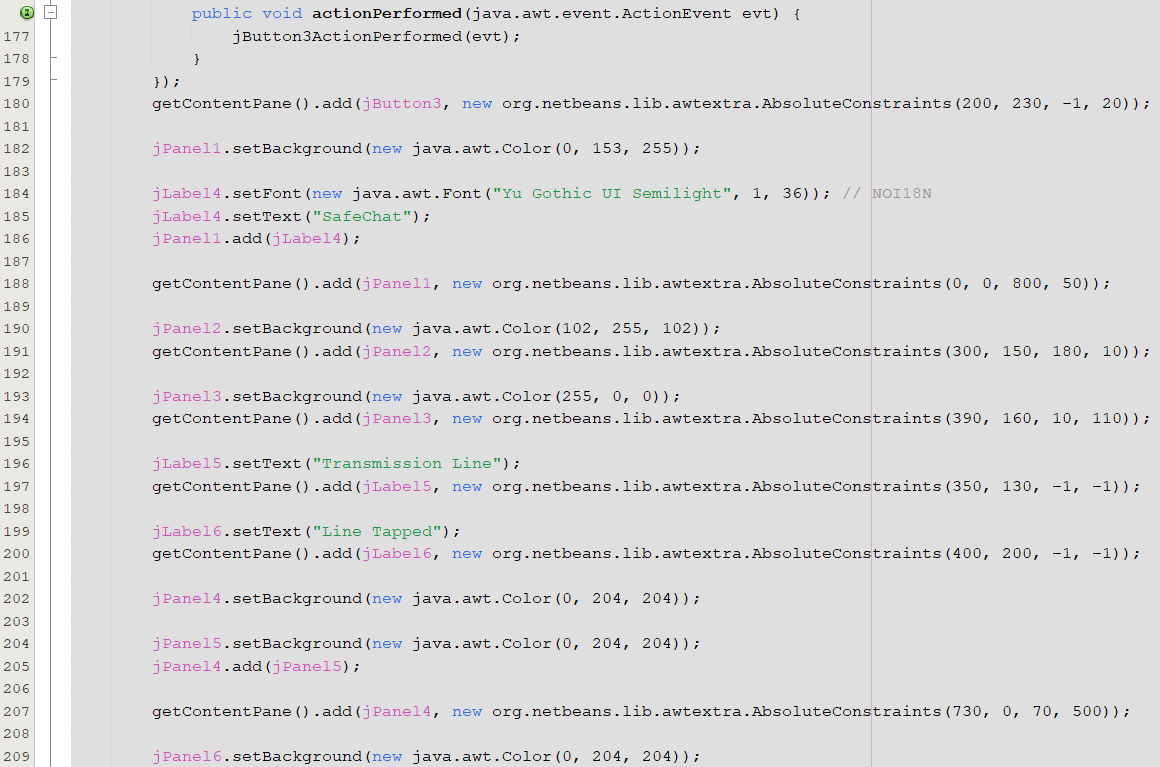


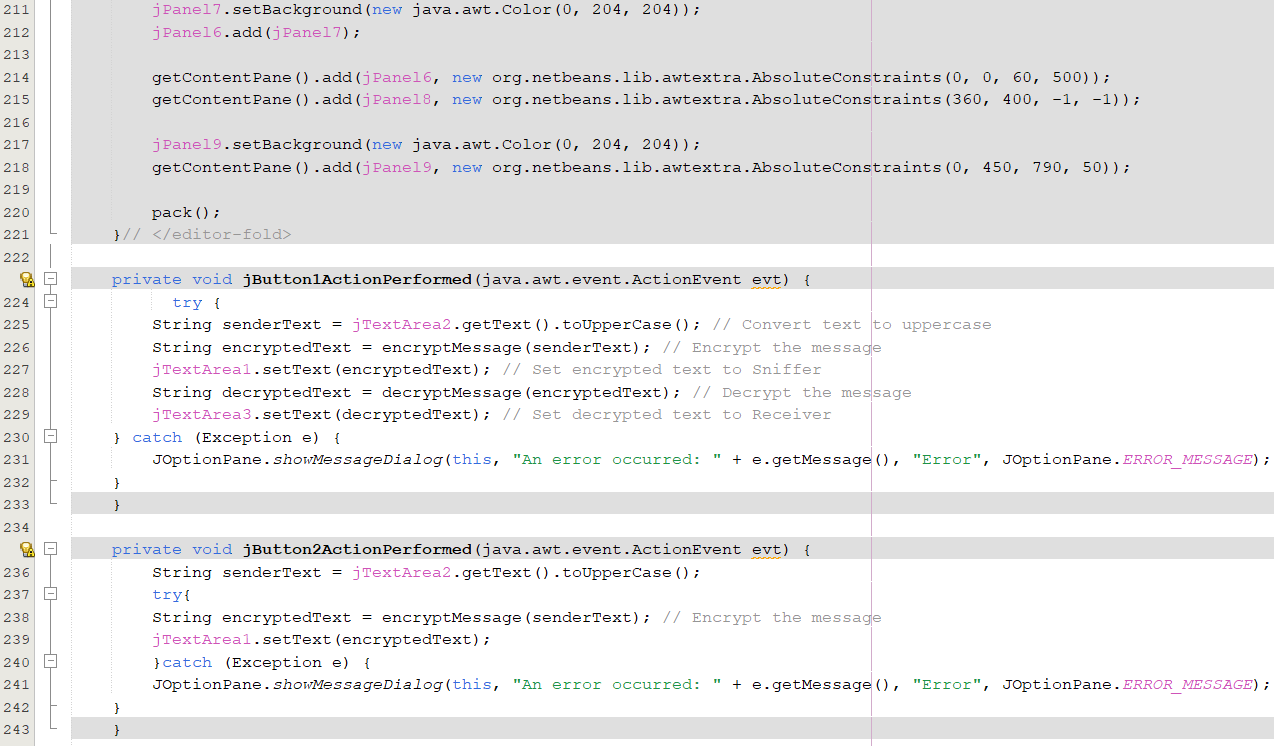


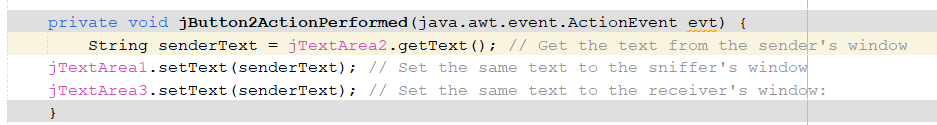




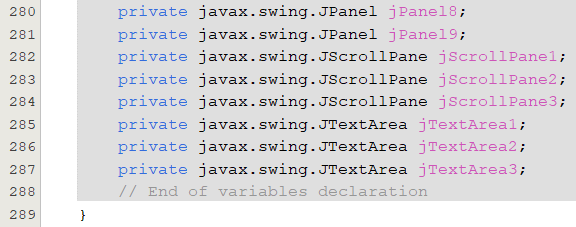




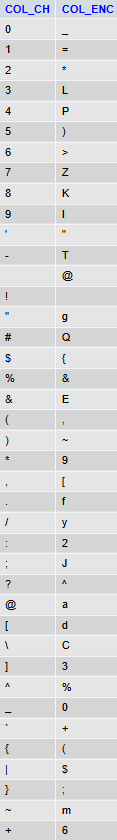








5] Database



5.1] Database Explanation

The database is designed in such a way that every ASCII Character has a unique ASCII Character that is been assigned to it.

The comparing and the retrieval of the data is based upon the corresponding values that are present in the respective fields.

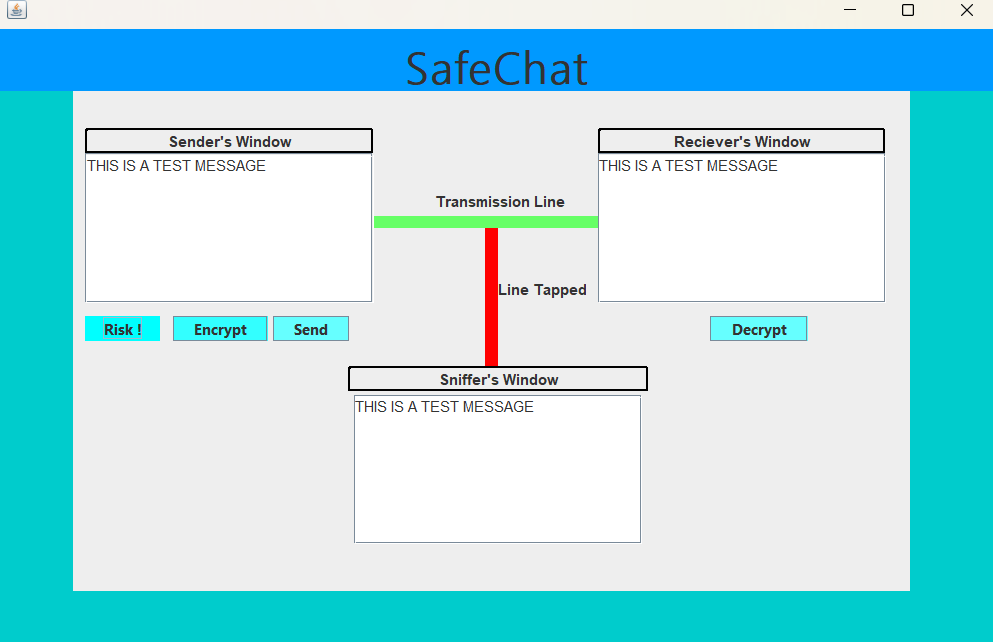
This is helping in encrypting the data into a non-readable format and also vice versa.

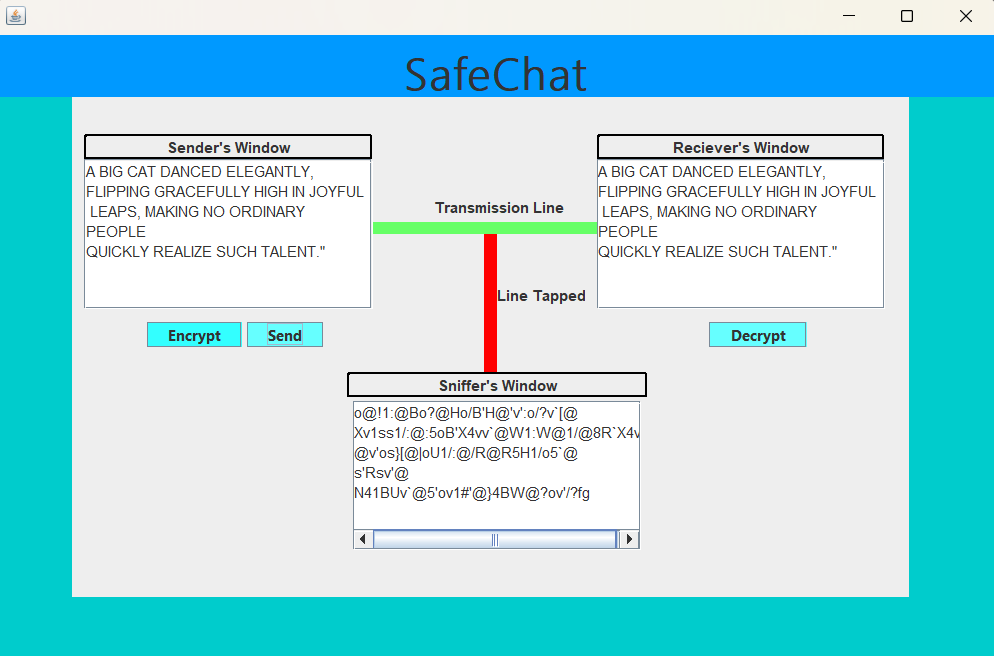
6] Explanation of the Code

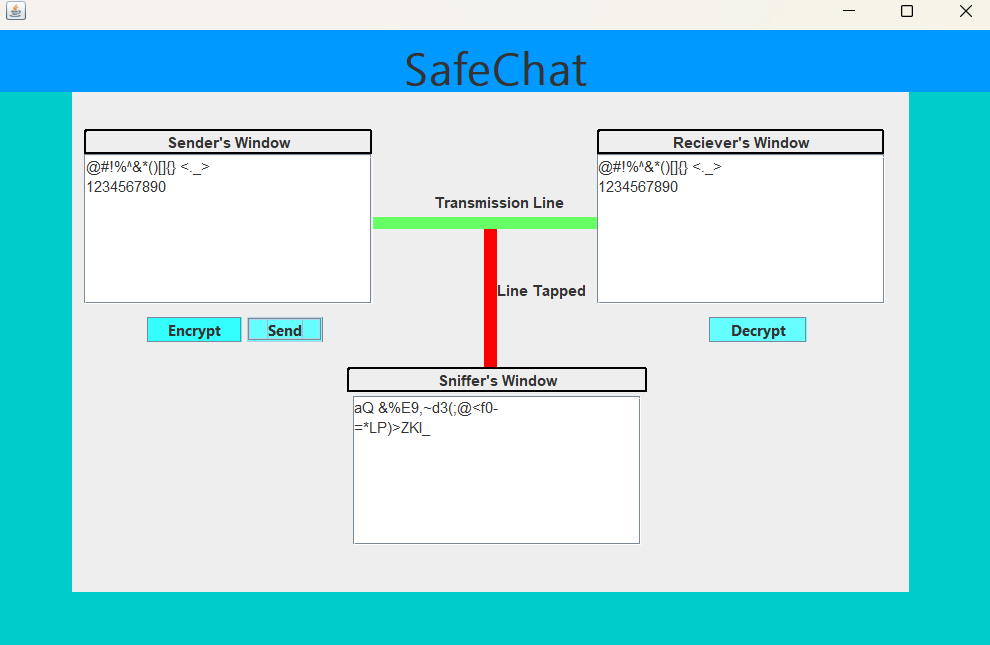
This Java program presents a simplistic messaging system with encryption and decryption qualities based on Swing for the interface and MySQL for database operations. A fundamental message system is implemented in this application by breaking it down into three windows: The Sender's Window, a field where a message is typed; Sniffer's Window, as such, showing what was intercepted in an encrypted form; and Receiver's Window, where the message is decrypted and can be read. It queries a MySQL database table called T\_ED, that holds all the mappings for character encryption and decryption, and, thus encrypts a message by substituting each character of the sender's message with its encrypted counterpart. The decrypt will simply retrieve the original characters from the encrypted message, doing nothing but the reverse mapping.

The application connects to a local MySQL database through JDBC. Prepared SQL queries are then used to implement the logic of encryption/decryption. For the encryption process, it maps each character to its corresponding encrypted value in the T\_ED table, while decryption is mapped from encrypted characters back to their original values. Still in relation to this application, it handles errors like database connections that failed by using JOptionPane dialogs. The UI elements of this program comprise JTextArea for message input/output, and buttons to start the action for encryption, decryption and sending-a very interactive tool for safe messages.

7] Result



This is the result when the data is sent without encryption. As is can be seen the Sniffer can clearly read the data. 

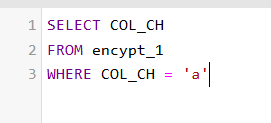
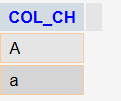
As it can be seen that all the Characters are successfully been encrypted and decrypted without the sniffer getting the understanding of the original message.

Even the use of numbers and special characters is able to pass through successfully.

8] Scope of Improvements

8.1] Restriction to all Capital message transfer

Since the database is unable to distinguish between the case sensitivity of the message must be sent in Capital letter. It can be seen in the following command ran in the SQL.





Here the expected return should be 1. However the inability of MySQL to differentiate between the case sensitivity, the retrieval result is 2.

8.2] User Input Validation

Currently, there is little to no validation on the user input before processing the messages. Adding checks to ensure that the input is not empty or does not contain unsupported characters could prevent potential issues during encryption and decryption. Implementing such validation would enhance the application's robustness and user experience.

9] Conclusion

In conclusion, the Java messaging application effectively demonstrates a practical implementation of message encryption and decryption using a graphical user interface and a MySQL database. The encryption is performed by using the logic of Mapping. By leveraging Swing for user interactions and JDBC for database connectivity, the application allows users to securely send and receive messages, showcasing the fundamental principles of data security in communication. Although the application meets its primary objectives, there are opportunities for enhancement in areas such as restrictions free input messages, error handling, input validation, and code modularity. By addressing these aspects, the application can become more robust and user-friendly, ultimately providing a more secure and efficient messaging experience. Overall, this project serves as a foundational step toward understanding secure communication practices and can be further expanded to incorporate advanced encryption techniques and additional features.

10] References

1. [Lesson: JDBC Introduction (The Java™ Tutorials > JDBC Database Access) (oracle.com)](https://docs.oracle.com/javase/tutorial/jdbc/overview/index.html)
2. [Mapping, encrypted and decrypted points for "cryptography". | Download Table (researchgate.net)](https://www.researchgate.net/figure/Mapping-encrypted-and-decrypted-points-for-cryptography_tbl2_267248088)
3. [What Is Data Encryption: Algorithms, Methods and Techniques (simplilearn.com)](https://www.simplilearn.com/data-encryption-methods-article)
4. [MySQL Tutorial (w3schools.com)](https://www.w3schools.com/MySQL/default.asp)
5. [Exception Handling in Java Tutorial (youtube.com)](https://www.youtube.com/watch?v=1XAfapkBQjk)
6. [Overview (Java SE 11 & JDK 11 ) (oracle.com)](https://docs.oracle.com/en/java/javase/11/docs/api/index.html)